TEST PERFORMANCE MEASUREMENTS

Release Date: December 20, 2000

Contract Number: GS-35F-4919H
Order Number: T0000AJ3739

CLIN #008-1

Prepared for:

U. S. DEPARTMENT OF EDUCATION
OFFICE OF STUDENT FINANCIAL ASSISTANCE (SFA)
400 Maryland Avenue, SW
Washington, DC 20202

Prepared by:

CTGi
Suite 250
10461 White Granite Drive
Oakton, VA 22124

This page left intentionally left blank

FOREWORD

CTG, Incorporated (CTGi) would like to thank the following personnel whose dedication and involvement made the development and completion of this document possible.

Kelly Conboy

Candace Jones

Frank Majewski

Annette Mazie

Paul Stocks

Sandra Stocks
Director Of Operations,
Washington Metropolitan Area
CTGi

This page intentionally left blank

EXECUTIVE SUMMARY

This document was prepared for the United States (U. S.) Department of Education, Office of Student Financial Assistance (SFA) by CTG, Incorporated (CTGi) for the purpose of developing a standardized procedure to provide Project Managers (PM) with the software information necessary to make informed decisions that impact project cost, schedule, and technical objectives.

This document, along with those listed below, will be integrated into and become the final deliverable of this contract, U. S. Department of Education, SFA, System Integration and Testing (SI&T) Process Handbook. The U. S. Department of Education, SFA, SI&T Process Handbook will then become integrated into the overall U. S. Department of Education, SFA Modernization Technology Handbook. The remaining documents that will comprise the SI&T Process Handbook are:

- System Integration and Testing Standards
- Procedures and Templates for Test Creation
- Procedures and Templates for Test Execution, Test Evaluation, and Error Correction
- Procedures and Templates for System Configuration Management (CM) and Quality Assurance (QA)
- Procedures For Using Testing Tool

Each of the above listed procedures, templates, and guide were prepared for delivery as a separate document.

All SI&T guidelines and procedures are focused on supporting systems or projects used in the development and execution of a comprehensive integration and testing program. To this end, this document contains information on understanding issues associated with procedures relating to system test management, system test planning roles, system testing organization responsibilities, collection of information related to progress of the system test phases, and required system testing practices.

This page intentionally left blank

Table of Contents

Table of Contents (Cont'd)

LIST OF EXHIBITS

This page left intentionally left blank

LIST OF ACRONYMS

CASE	Computer Aided Software Engineering
CM	Configuration Management
CMO	Configuration Management Office
COTS	Commercial Off the Shelf
CTGi	CTG, Incorporated
DCR	Document Change Request
CSCI	Computer Software Configuration Item
GUI	Graphic User Interface
HWCI	Hardware Configuration Item
IDE	Interactive Development Environment
IPT	Integrated Product Team
IRS	Interface Requirements Specification
LOC	Lines of Code
PM	Project Manager
POC	Point of Contact
QA	Quality Assurance
QAO	Quality Assurance Office
QC	Quality Control
SD	System Development
SDF	Software Development File
SFA	Office of Student Financial Assistance
SI&T	System Integration and Testing
SPR	System Problem Report
SQT	System Qualification Test
SRD	Software Requirements Document
SSS	System or Subsystem Specifications
STD	System Test Description
STO	System Test Organization
STP	System Test Plan

LIST OF ACRONYMS	
STR	System Test Report
SU	Software Unit
TM	Test Manager
TPM	Test Performance Measurements
TRR	Test Readiness Review
WBS	Work Breakdown Structure

INTRODUCTION

Background

The United States (U.S.) Department of Education, Office of Student Financial Assistance (SFA) organization, contracted CTG, Incorporated (CTGi), in August 2000, to develop standardized System Integration and Test (SI&T) procedures. These procedures will be used for guidance, planning, and implementation involving current and future U. S. Department of Education enterprise information technology systems projects.

Objective

This document describes the procedures, activities, and tasks necessary to ensure the U. S. Department of Education, SFA, a consistent approach for measuring and defining system integrity, reliability, efficiency, and functionality to ensure successful information technology system integration and testing tasks and activities.

During the SI&T process, the focus is on preparing the system for delivery to the customer. This usually means that the focus is on evaluating system quality. SI&T is often one of the shortest and most intensive activities. Consequently, the Test Performance Measurements (TPM) process must focus on providing rapid data collection, analysis, and feedback to project management so that effective decisions can be made in a timely manner. On many projects, this results in increased analysis of documentation and various reports. A weekly reporting interval for System Problem Reports (SPR) is often used during the SI&T phases. In some cases, daily test progress and SPRs are provided.

The determination of the reporting interval depends on many factors, but there is usually an increase in measurement activity during SI&T process. Effective TPM gives an accurate and comprehensive assessment of testing activities, minimizes and standardizes the burden of data collection, and is accepted and used to improve SI&T process performance.

Reference to subjects, actions, and consequences made in this document, outside of the scope of the SI&T process, are examples of how the TPM process and associated information can impact other aspects of system life cycle engineering and development.

Applicability

When the SI&T process is performed by either the U. S. Department of Education, SFA, staff and/or contractors, this document applies, unless specifically excluded, in the program/project plan, contract, etc. This document is used for the creation of guidelines and procedures for the planning, preparation, execution, analysis, and evaluation, of all types of U. S. Department of Education, SFA, information technology project integration and testing.

Document Organization

This document contains five narrative sections, a Glossary, a Bibliography, and one appendix. Section 1, Introduction, provides brief background information and states the guiding objective and applicability for the document. Section 2, Proposed System Test Organization Overview, defines a proposed System Test Organization (STO) that can be used during the SFA SI&T process. Section 3, Test Performance Measurements Overview, is an overview of the TPM process and defines goals and process participant roles. Section 4, Tailoring Test Performance Measurements, establishes the guidelines necessary to identify measurement requirements to address specific project issues. Section 5, Effective Use of Measurement Data, presents an example of information gathering for the TPM process, with analysis and utilization during the SI&T process. Appendix A, Measurement Tailoring Plan, presents an example of how a measurement tailoring plan is created.

PROPOSED SYSTEM TEST ORGANIZATION OVERVIEW

The adaptation of a system testing discipline for the production of software-intensive systems requires more than just an understanding of the technical issues. Developing a testing capability starts with management establishing system test policies that define software testing. An additional concern of management is coordinating software test activities to support the needs and priorities of the process, satisfy the customer, and achieve the overall objectives of testing. To that end, responsibility for testing activities must be established. The following paragraphs discuss a proposed STO structure. Figure 2.1 presents this proposed organization.

Error! Not a valid link.

Figure 2.1 Proposed System Test Organization

The STO structure outlined in this section is based on the implementation of developed, standardized procedures for conducting the SI&T process. There must be documented processes established, which provides members of the STO with information for interpretation and execution of SI&T process standards. Prescribed practices and procedures should encompass administrative, analysis, planning, review, development, integration, testing, documentation, CM, and QA. A well-organized STO assists in ensuring that the following requirements are met:

- Test planning, design, integration, execution, and evaluation responsibilities are explicitly assigned.
- Test objectives and phases of tests to be conducted are documented in a System Test Plan (STP).
- Test case(s) and test case procedure(s) are developed for all formal testing phases and documented in a System Test Description (STD).
- Test resources and artifacts are under CM control.
- Testing activities and documentation are reviewed and evaluated by QA.

The development of test policies and standard test processes are performed or coordinated by designated personnel. Coordination of the testing process ensures that activities are performed, documented, reviewed, and approved in accordance with approved test policies and standards throughout the testing life cycle.

Executive Manager

The executive manager is generally an organizational or enterprise manager responsible for multiple projects. This manager defines high-level performance and business objectives and ensures that individual projects support the overall organizational strategy.

Project Manager

The PM oversees all personnel, technical, quality, cost, and schedule aspects of the SI&T process. In this context, the PM has the overall responsibility for the completeness of development and testing efforts. The PM is the principle point of contact (POC) for the entire STO, including CM, QA, testing, and development groups. The PM establishes the direction and management controls to ensure the success of all project activities and the resolution of issues that may inhibit the SI&T process.

Within the STO, depending on the size and scope of the project, the PM will meet with various personnel from supporting organizations. These personnel include the Executive Manager, Configuration Management Office (CMO), and Quality Assurance Office (QAO) and they plan, prioritize, and coordinate testing tasks and procedures within their organizations. In this way, standards and procedures developed for the project life cycle are incorporated into the SI&T process. The PM then has the flexibility to assign tasks to personnel with appropriate expertise.

Quality Assurance Group

QA is the process that evaluates the form, structure, and/or compliance of a system in relation to applicable standards. QA provides standards that ensure all Quality Control (QC) requirements are implemented during the SI&T process. The QA group provides evaluation disciplines to the SI&T process.

The QAO is the officiator of the QA system life cycle process. The QAO develops standards and procedures used to implement QA functionality throughout the SI&T process. Although the QAO is responsible for reviewing and auditing the entire system project, this subsection only discusses QA as it relates to SI&T and the form, structure, and/or compliance to the SI&T processes.

The QA group reports directly to the PM. The QA group provides the PM with the assurance that all QC requirements for SI&T process are met by conducting reviews and/or audits of testing activities. The QA group creates both formal and informal reports from the results of the reviews and/or audits. Throughout the SI&T process, the QA group provides the following information, to be used in the TPM process:

- Evaluation of the system/test corrective action process.

- Evaluation of the CM process.
- Evaluation of all system/test documentation.
- Evaluation of all guidelines, test cases, and test case procedures.
- Creation of measurement information.

Configuration Management Group

CM controls the integrity of the project being developed and tested. CM is the vehicle that manages and controls changes/modifications made to documentation, software, and environments during the life cycle of the SI&T process. The CM group provides the disciplines that apply technical and administrative direction and control to the SI&T process.

The CMO is the officiator of the CM system life cycle process. The CMO develops standards and procedures used to implement CM functionality throughout the SI&T process.

The CM group reports directly to the PM and executes baseline control, version control and identification, and change control standards and procedures throughout the SI&T process. The CM group creates both formal and informal reports based on progress of the testing life cycle. Throughout the SI&T process, the CM group provides the following control over information used in the TPM process:

- Ensure that all formal software, hardware, and test environment configurations, as specified in the Software Requirements Document (SRD) and System Test Plan (STP), are placed under CM control.
- Ensure that all formal documentation is placed under CM control.
- Ensure that formal release procedures for CM approved software, documentation, and test environment versions are established.
- Ensure prevention of unauthorized changes to controlled software, documentation, and test environment, and ensure incorporation of all approved modifications/ changes.
- Track and report system problems.
- Create measurement information.

System Development Group

The System Development (SD) group interacts directly with testing group functions; however, it remains a separate entity within the STO. This separation ensures the objectivity of the testing group, when evaluating system requirements and quality.

Within the realm of testing, the SD group is responsible for the analysis, design, testing, implementation, quality, and documentation of the Software Unit (SU) Test phase testing. The SD group reports schedule, quality, technical performance, and documentation of the SU Test phase effort to the PM. In addition, the SD group ensures that policies and procedures established for the Software Development Files (SDFs) are followed and provides comprehensive SU Test phase testing coverage, which will contribute to the overall success of the testing process.

During the Integration Test, Performance Test, and System Qualification Test (SQT), the SD group is responsible for the repair of all software problems reported and cooperates with the testing group to ensure the reported problems are solved and tested in an expeditious manner.

The SD group provides the PM with requested measurement information regarding all efforts, activities, and responsibilities during the SI&T process.

Test Manager

The Test Manager (TM) manages and guides all integration and testing functions of the SI&T process. The TM will specify test standards, allocation of resources, test scheduling, and management of the test engineers. Additional TM responsibilities include:

- Develop software test process standards in a concise and usable form and the process by which the standards are communicated to other groups of the STO.
- Create software test methodologies.
- Format specified standard tools and technologies that support the SI&T processes.
- Schedule of time and resources, for both hardware and software testing.
- Resolve issues that may obstruct or inhibit the testing schedule or effort.
- Collect all measurement information requested by the PM.

Test Engineer

The test engineer(s) is responsible for testing the system during the SI&T process. The test engineer(s) is responsible for development and creation of the STD. It is the responsibility of the

test engineer(s) to ensure STP and STD tasks are implemented during the SI&T process. The test engineer(s) performs the Integration and Performance Test phase testing. During the SQT phase, the test engineer(s) may or may not perform the testing. If, during the SQT phase, the test engineer(s) do not perform the testing, they act as witnesses of the testing. As the test case procedures are executed, the test engineer(s) communicates the status of testing to the TM.

Test engineer(s) author the System Test Report (STR) at the end of Integration Test, Performance Test, and SQT phases of the SI&T process. The test engineer(s) documents problems encountered during testing by creating SPRs. It is the test engineer's responsibility to re-test SPRs after they are returned from the SD group and ready for re-test. After re-testing, the test engineer(s) recommends closure of the SPR or returns the SPR to the SD for further investigation and repair(s).

This page intentionally left blank

TEST PERFORMANCE MEASUREMENTS OVERVIEW

Measurement is a key element of successful management in every well-established engineering discipline. TPM present an approach for tailoring and implementation of an effective measurement process for software-intensive projects. The objective is to provide PMs with the system information required to make informed decisions that impact project cost, schedule, and technical objectives.

TPM describe system measurement as a systematic but flexible process that is an integral part of the overall project management structure. Project issues drive the TPM process. The process is adaptive to meet the specific information needs and characteristics of each individual project. The process is based on a proven set of system measurement principles derived from actual experience on government and industry projects. These principles represent measurement "best practices" and make the TPM process an effective management tool, not just another project management "requirement."

The TPM process provides a foundation for objectively managing the technical and acquisition aspects of a software-intensive project. The process, implemented either as a stand-alone discipline or integrated with project risk management and financial performance management techniques, establishes a basis for informed decision making and communication throughout the STO.

TPM outlines and provides guidance for processes and procedures necessary for consistent construction of appropriate measurement plans. Appropriate measurement plans are based on the availability of the documentation necessary for the assessment, reporting, and tracking of information technology SI&T projects and tasks.

Measurement Implementation Roles

The TPM process is an integral part of the system and/or software testing process. Many members of the STO play important roles. Appropriate resources must be allocated for the TPM process to work effectively.

The most important roles in the TPM process are as follows:

- **Executive Manager.** The executive manager is generally an organizational or enterprise manager responsible for multiple projects. This manager defines higher-level performance and business objectives and ensures that individual projects support the overall organizational strategy. TPM helps the executive manager determine the status of individual projects and make decisions that apply across the organization.

- **Project Manager.** The PM is responsible for identifying issues, reviewing analysis results, and acting on measurement information. In the optimal case, both the system development and testing group will have PMs who use TPM information to make decisions for their respective group and to communicate objectively between the groups.

- **System Test Organization.** The STO is responsible for the day-to-day development and testing of a system or software application. The SD, Test, QA, and CM groups can be comprised of both government and industry personnel or organizations, and may be defined within an Integrated Product Team (IPT) structure. STO groups are responsible for collection of measurement data on a periodic basis and all groups use the measurement results to guide SI&T activities.

- **Measurement Analysis Personnel.** The measurement analysis role can be assigned to an individual or a team. Responsibilities include developing the project TPM plan, if necessary, collecting and analyzing measurement data, and reporting results throughout the STO. Each organization within the STO that makes critical system decisions should have an independent measurement analysis capability.

Ensure that all participants involved in the TPM process understand and commit to their responsibilities. This ensures that accurate information is available to support effective communications and informed decision making.

Communication

The TPM process is used as a basis for objective communication. Measurement activities should not be conducted in isolation by any of the groups that comprise the STO. The PM should communicate with the entire STO at each step of defining TPM requirements and analyzing measurement data. Most decisions based on the measurement data will affect more than one group. For example, a corrective action that is identified and planned in cooperation with the SD group is more likely to succeed than one that appears to be arbitrarily imposed by the system test manager.

While there may be some differences between the issues of concern to the SD group and the Test group, there should also be a high degree of commonality. It is important to ensure that all parties use the same data and have a common understanding of data definitions and know what the data represents.

TAILORING TEST PERFORMANCE MEASUREMENTS

The first part of the TPM process focuses on identifying the measurement requirements to address specific project/task issues. This process includes identifying project issues, selecting and specifying appropriate measures, and integrating the measures into the SI&T process. The first part of the TPM process and its component activities are discussed in this section

The remainder of the TPM process, application of TPM and required measures needed to support TPM, is a very interpretive process. TPM may be deeply interwoven into the project, or it maybe a "stand alone" adjunct to the project. The application of TPM maybe a formal process or an informal process, depending on the complexity and size of the project. A fictional example that follows measurement requirements to address specific project/task issues of the TPM process is included in Appendix A, and provides an example of "one of many" procedures for applying TPM.

Test Performance Measurements Tailoring Overview

This section outlines the process for defining TPM requirements and developing an appropriate measurement plan. The objective of the measurement tailoring process is to define the set of measures that provides the greatest insight at the lowest cost. The tailoring process focuses effort and resources on obtaining information regarding high priority issues first.

Project issues drive the entire measurement process. The issues determine which measures are selected, how measurement results are analyzed, and how managers make their decisions.

Figure 4.1 depicts the measurement tailoring process.

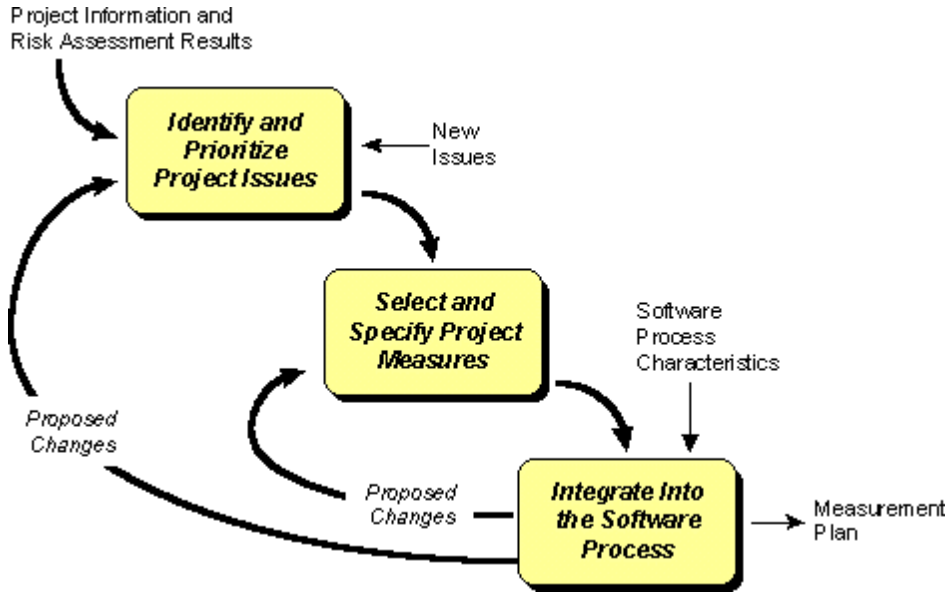


Figure 4.1 Measurement Tailoring Process

The first activity in the tailoring process is the identification and prioritization of specific project issues. Project issues are derived by reviewing project information, such as objectives, constraints, technical strategies, estimates, and risk analysis results, as well as general organization requirements. The basic concern in this activity is identifying system measures that have the greatest potential impact on the project.

The next tailoring process activity is to define appropriate project-specific measures. These measures are selected by applying defined measurement tailoring mechanisms of common system issues, measurement categories, and measures. The basic objective in this activity is to select measures most appropriate to the project issues.

The final tailoring activity is integrating the measures into the SI&T process. The system environment, development approach, and management process affects the definition, availability, and utility of the desired measures. Existing measurement implementations, if any, should be considered for their applicability to the project information requirements. Results of this integration are documented in a Project Management Plan. This plan may be formal or informal, depending on the nature of the project.

Figure 4.1 shows that the tailoring process is iterative. New issues may be discovered or refinements may be proposed in the course of examining the SI&T process. Alternative measures may be proposed to satisfy project office information needs, while minimizing cost. Tailoring may also occur after the initial Project Management Plan is developed. New issues and new opportunities for measurement may be discovered as the project matures and previously identified project issues may decrease in importance.

Identify and Prioritize Project Issues

An effective measurement process helps the PM recognize and deal with problems and risks that might prevent the project from being successful. TPM refers to these obstacles as issues. TPM tailoring process begins with the identification of project-specific issues.

The shaded area of Figure 4.2 shows the detailed tasks that comprise the identification and prioritization of project issues. Potential issues are identified using all available project information. This task can be either formal or informal, and should address the concerns of all STO groups involved in the project. The results of the project risk assessment process, if any, should also be integrated into the issue identification task. Identified project issues are mapped to TPM common system issues. This mapping of project issues to common system issues helps in the selection of appropriate measures for each issue from the Table 4.1, Measurement Categories and Related Questions. New or revised issues and measures may be defined to best support the unique aspects of a specific project. Finally, project issues are prioritized. The priority assigned determines the emphasis placed on measuring and tracking the issue through the measurement process.

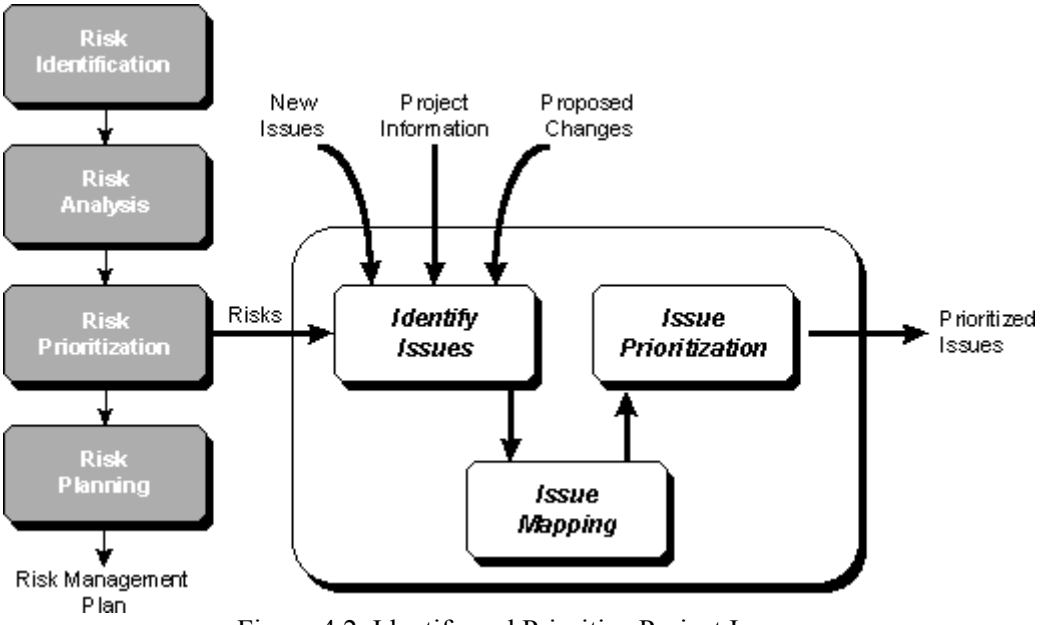


Figure 4.2 Identify and Prioritize Project Issues

Issues

Issues are areas of concern that may impact achievement of a project objective. Issues include problems, risks, and lack of information. These terms are summarized below:

- A problem is an area of concern that a project is currently experiencing or is relatively certain to experience.

- A risk is an area of concern that could occur, but is not certain.

- A lack of information is an area where the available information is inadequate to reliably predict project impact.

Identifying something as an issue does not necessarily mean that it is a problem. In fact, identification of issues and careful tracking minimizes the potential of serious problems occurring that could negatively impact project progress and/or success.

In addition to issues identified at the start of the project, new issues may arise as the project progresses. New or evolving requirements, changes in technology, and other factors usually result in the identification of derived issues as the project progresses. Consequently, the tailoring process usually needs to be revisited periodically during the project life cycle.

Risk Management

Risk management is a discipline separate from TPM that is instrumental to the realization of success during a software-intensive project. The risk management process is implemented in parallel with the TPM process and interfaces with the TPM process directly. Risk management consists of two primary activities: risk assessment and risk management. Risk assessment helps identify, analyze, and prioritize project risks. Risk management focuses on planning, monitoring, and controlling identified project risks. Both activities work in conjunction with the measurement process.

The risk assessment process is closely aligned with measurement tailoring. As depicted in Figure 4.2, risk assessment delivers formally defined and prioritized risk information into the measurement tailoring process and specifically supports project issue identification. Risk assessment may point to potential issues with requirements, technology, process, cost, or schedule. Even if a formal risk assessment is not performed, issues can still be identified. The measurement analyst should also understand that not all risks are quantifiable and that not all issues are risks. Therefore, risk techniques alone may not be adequate to effectively tailor a measurement process.

Risk assessment results typically include a list of risk items that are quantified in terms of their significance. Two dimensions for risk quantification are:

- Probability -- How likely is it that a risk will result in a problem?

- Impact -- How much impact is the potential problem likely to have on project success?

The numeric value of probability and impact information is commonly referred to as risk exposure and is used to help prioritize the identified risks.

Risk assessment results are a primary input to the project risk management plan. It is important to understand the behavior of project risks. Overall exposure of a given risk to the project can change based on when it is expected to occur and where it is expected to be applicable within the project STO. Identified project risks tend to change over time and are influenced by project events.

Risk management and measurement are synergistic. Both disciplines emphasize the prevention and early detection of problems rather than waiting for problems to occur or become critical. The risk management process helps identify and prioritize system issues. The measurement process helps quantify the likelihood of a risk occurring and the amount of potential impact. Risk management usually addresses more issues than can be quantified using measurement. For example, environmental and political risks are included in the risk management process but are not generally relevant with respect to system measurement.

Select and Specify Project Measures

This section describes the activity in the TPM tailoring process that helps select the best set of measures to address the identified project issues. Since every project is described by a unique set of issues, the measurement requirements for each project are also unique.

The tasks in the measurement selection activity are depicted in Figure 4.3. These tasks include identification of appropriate measurement categories for the identified issues, selecting the most appropriate measures within the categories, and specifying data requirements to define and implement measures. These tasks are discussed in more detail in the following sub-sections.

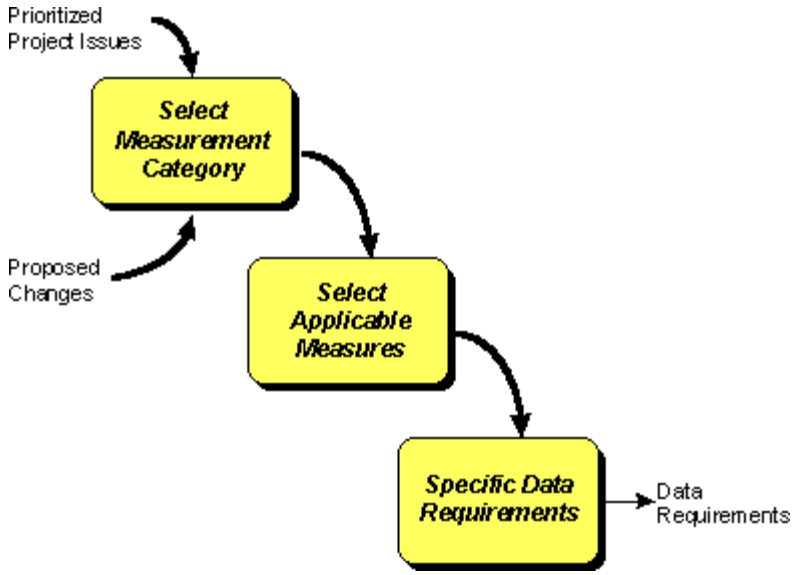


Figure 4.3 Measurement Selection and Specification

Although the issues that must be addressed primarily drive measurement selection, the overall characteristics of the project and its software development approach should also be taken into consideration. The types of graphs in reports (i.e., indicators) produced during analysis also affect measurement choices. Anticipating the types of graphs and reports that will help define required measures and data attributes is necessary.

Select Measurement Categories

The first task in the selecting and specifying project measures activity is to select the measurement category that best addresses the identified issue. If several issues are similar, then the same measurement category may suffice for those issues.

One method of determining whether or not a category matches an issue is to consider the types of questions the measures in that category answer. Table 4.1 provides questions corresponding to TPM categories. This table can be used to determine the measurement category or categories that most closely align with a specific project issue.

Issue	Measurement Category	Question Addressed
Schedule and Progress	Milestone Performance	Is the project meeting scheduled milestones? Are delivery dates slipping?
	Work Unit Progress	How are specific activities and products progressing?
	Incremental Capability	Is capability being delivered as scheduled, in incremental builds and releases?
Resources and Cost	Personnel	Is effort being expended according to plan? Is there enough staff?
	Financial Performance	Is project spending meeting budget and schedule objectives?
	Environment Availability	Are necessary facilities and equipment available, as planned?
Growth and Stability	Product Size and Stability	Are the product size and content changing?
	Functional Size and Stability	Are the requirements and associated functionality changing?
Product Quality	Problems	Is the system "good enough" for delivery to the customer? Are open problems being closed?
	Complexity	Is the software testable and maintainable?
	Rework	How much additional effort is being expended due to changes and errors?
Development Performance	Process Maturity	Will the developer be able to meet project constraints? Is the developer likely to succeed given past performance?
	Productivity	Is the developer efficient enough to meet current commitments?

Issue	Measurement Category	Question Addressed
Technical Adequacy	Target Computer Resource	Utilization: Is the target computer system adequate? Is computer hardware suitable and available for expansion?
	Technical Performance	Are project requirements, such as response time and accuracy, being met?
	Technology Impacts	Is the planned impact of the leveraged technology, such as common architectures and Commercial-Off-the-Shelf (COTS), being realized?

Table 4.1 Measurement Categories and Related Questions

As an example, consider the common software issue of Schedule and Progress. Three measurement categories, Milestone Performance, Work Unit Progress, and Incremental Capability are mapped to this issue. The measures in these categories address schedule- and progress-related concerns, but they do so with different types of information and at different levels of detail.

Milestone performance measures provide basic start and end dates for defined software activities and events. This is adequate for developing and reviewing Gantt schedules; however, the measures do not address the degree of completion of individual software activities and products at any point in time. More detailed schedule and progress information is provided by the measures in the Work Unit Progress measurement category. Lastly, the measures in the Incremental Capability category show whether or not software components or functions are being completed, as planned, for each version or release in an incremental software development approach.

For example, if the project-specific issue is "progress of COTS software integration," then the work unit progress category is appropriate because the issue involves a question about the progress of a specific activity, namely integration. If the project-specific issue is "budget overruns to fix unanticipated problems," then the rework category is pertinent because the issue concerns the extra effort applied to correct latent defects.

Always choose the measurement category that provides the best fit for the prioritized list of issues. For critical or high-priority issues, consider selecting more than one measurement category. This will lead to different measures and measurement information, allowing for more in-depth analysis.

Select The Applicable Measures

The second task in the activity of selecting and specifying project measures is to choose measures that best address the specific project issue(s). The overall objective is to define measures that adequately address the identified issues and are practical to implement, given the management and technical characteristics of the project.

A number of measures may apply to one issue. In most cases, it is not practical to collect all or even most of the possible measures for an issue. Generally, more measures should be collected to track high-priority issues. Identifying the "best" set of measures for a project depends on a systematic evaluation of potential measures with respect to issues and relevant project characteristics.

For example, if "growth and stability" is selected as an issue, requirements and software size measures will be used to track it. The appropriate measure will depend on the nature of the project. Language type and application domain influence the choice of a size measure selected. Information systems may use function points to measure size, while other systems may use lines of code.

Once a measurement category is selected, a measurement selection criterion can be applied to identify the best measures for the project. Measures are selected based on the following criteria:

- **Measurement effectiveness.** How effective is the measure in providing the desired insight? Is it a direct measure of the software characteristic in question? Does the measure provide insight that relates to more than one issue?

- **Domain characteristics.** Are certain measures more likely to be used in a given domain? For example, response time is widely used to measure target computer resource utilization in information systems, while memory utilization is more widely used in Unix, Windows New Technology (NT), or Graphical User Interface (GUI) applications.

- **Project management practices.** Can existing management practices be leveraged to support the measurement requirements? For example, is a scheduling system in use that provides one or more of the desired measures?

- **Cost and availability.** What data should be readily available within the context of the project? How much effort will be required to extract and package the data for analysis? For example, extracting data using electronic sources usually costs less than manual collection.

- **Life-cycle coverage.** Does the measure apply to the life-cycle phase under consideration? Does it apply to multiple life-cycle phases?

- **External requirements.** Has the overall organization or enterprise imposed any related measurement requirements.

- **Size/origin of software.** Does the size or scope of the software project justify a larger investment in measurement? Does this measure make sense for this type of software, such as COTS?

In most cases, the selection activity requires that tradeoffs be made among the measurement selection criteria. For example, a given measure may directly address a high-priority project issue, but may be too costly to implement in terms of time and resources. Some measures, when used in conjunction with other specific measures, support multiple analysis needs. For example, lines of code are used to calculate and analyze software development performance in terms of productivity and quality. This measure may therefore be important even if Growth and Stability is not a priority issue.

In general, measures from different measurement categories within the same common software issue can be substituted with some degree of effectiveness. Also, measures that are categorized under different common software issues may provide additional insight into the issue in question. Obviously, it is better to use a substitute measure than to select a measure that cannot be implemented. After the initial measures are selected, they should be reviewed to ensure that the high-priority issues are addressed and that there is adequate coverage across all identified issues.

Specify The Data Requirements

Once the measures are selected, the last task of the measurement selection and specification activity can be performed. This is to specify the data requirements for each identified measure. The data requirements defined in this task become the basis for agreement with the developer to define what and how data will be provided. Within an IPT environment, specification should be done in conjunction with the developer. For an in-house or commercial development, the data requirements still need to be specified and defined; however, this may be done more informally in some cases.

The appropriate level of detail for the collection of measurement data must be defined. The frequency and format of data deliveries must also be specified, since data may be reported less often than collected by the developer.

To support the measurement analysis process, data must be collected at a level of detail that will allows isolation of problems. Some factors to consider in determining the appropriate level of data collection include:

- Requirements and size data are normally tracked at the CSCI level. Consider tracking size data at a lower level, if the CSCIs are large.

- Progress is normally reported at the level of major activity, such as design. Consider tracking at the level of sub-activities, if the schedule is long.

- Keep data from subcontractors separate, if the subcontractors have significant software development responsibility, or use a different development process.

- Maintain separate counts of size data for each language type, unless the languages are comparable.

- Maintain separate counts of size data, level of effort, and problem reports for each category of software, such as new development, reuse, and COTS, especially if project success depends on realizing some specific benefit from these approaches.

- Keep separate counts for each priority category of SPRs, especially if the project maintains a large backlog of problems.

In determining the appropriate level of detail, the cost of data collection, data processing, and analysis must be balanced against the need for detailed insight into project issues. More detailed data allows greater flexibility in analysis in terms of defining new indicators and determining the source of potential problems. However, a greater level of detail also implies a greater volume of data and a greater cost for the measurement process. More detailed data should be sought to track those issues considered to be most important. All of these recommendations for selecting measures and their level of detail must be tempered with an understanding of the developer process.

Selecting and Specifying Measures For Existing Projects

TPM selection and specification guidance is generally structured to support a sequential tailoring of the measurement process. In some instances, the need to implement a measurement process is driven by a significant project event or issue that must be supported by objective system information immediately. In other cases, new policy guidance or other external requirements, such as a major milestone review, make it necessary to implement measurement on a project that is underway.

The tailoring approach still begins with the identification and prioritization of specific project issues. In all likelihood, key issues are already identified and the immediate objective is to identify data that can be used to provide the PM with meaningful information. However, less emphasis should be placed on defining data requirements and more emphasis placed on identifying existing measurement opportunities. Successfully implementing measurement on an existing project means taking advantage of measurement opportunities present in the project software management and technical processes. Often the required data exists, but is not mapped to the issues or collected in any systematic way.

Integrate Measures Into The Software Development Process

Up to this point, the measurement selection process has largely been driven by "what" the PM needs to know about the issues. The next task is to look at "how" the measurement process will actually function within project management and technical processes. Data available from the developer may not map exactly into the ideal defined measurement requirements.

This final tailoring activity includes three tasks, as depicted in Figure 4.4. First, the software development process and environment are characterized. Next, opportunities for measurement within that environment are identified. Finally, measurement requirements are specified, typically in a Project Management Plan.

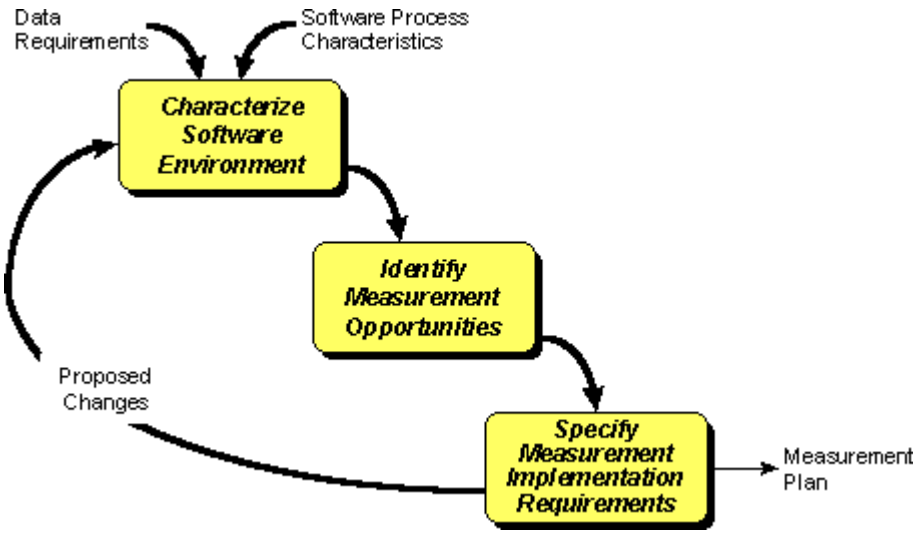


Figure 4.4 Integrate Measurement Into the Software Development Process

To better integrate the measures into the software development process during the course of performing these tasks, the developer should propose changes to project measurement requirements. The measures and data requirements selected in the tailoring activity form the basis for agreements between managers and the developer about the specific data elements to be provided for analysis. This agreement may be accomplished via a formal contracting process or a less formal understanding, in the case of internal developments. The result of this activity is a definitive statement of the measurement approach to be followed, often documented in a measurement plan or incorporated into the Project Management or Software Development Plans.

The tasks required to integrate measurement requirements into the software development process are discussed below.

Characterize Software Environment

The development process used by the developer defines how software is measured. The definition of a measurement process cannot be based solely on the objectives of the managers.

To collect measurement data in the most cost-effective and useful manner, the software development process of the developer must be considered. Project issues identify information the measurement process must derive from the collected data. The software development process determines what specific data items can be collected and how that can be accomplished.

One purpose of the measurement process is to provide insight into developer performance. Measures collected must objectively represent activities and products of the software development process. As much as possible, managers should select measures that are normally collected by the software developer. The software development processes employed by any software subcontractor should be considered before a decision is made on which measures are collected.

Some key factors to consider when deciding which measures to collect follow:

- The life-cycle model or activity structure used to define the software development process.
- The software product structure, including versions and releases defined by the developer.
- The product line architecture.
- Current measurement activities employed by the developer.
- Software technology, including programming language, design language, and tools.
- Planned source of the software; such as COTS, newly developed code, and reuse.
- Management, review, testing, and inspection practices employed by the developer.
- Engineering and management standards to be applied.

The software development process has major impact on the cost and effectiveness of the software measurement process. Whenever possible, current practices of the developer and existing data collection mechanisms should be used. New measurement requirements should be kept at a minimum. Use the project Work Breakdown Structure (WBS), including product structure and activities, as the basis for measurement.

To the extent that activities of the software development process are well-defined, measuring them provides useful information. An ad-hoc or ill-defined process makes it difficult to tell exactly what is being measured.

For many issues, the data available changes across life-cycle activities. For example, during SU Test phase, progress may be measured in terms of units designed and coded. During remaining SI&T phases, progress may be measured in terms of tests attempted and passed. The measurement analyst must ensure that relevant measures and indicators are provided throughout the project life cycle, and make modifications, as appropriate.

Before measurement requirements are finally agreed to, managers should understand the software development process and obtain direct feedback from the developer on project measures. The measurement process should not be used to force process changes on the developer. Giving appropriate consideration to the software development process helps ensure that useful data is provided with the lowest impact and cost.

Identify Measurement Opportunities

During measurement planning, high priority should be given to identifying any measurement mechanisms already in place within the development organization. This is especially important when implementing measurement on an existing project. Special attention should be given to databases and tools supporting project management, QA, and CM. Extracting and delivering data from electronic sources is usually more cost-effective than manual or paper forms-based collection methods.

Software data comes from many sources. Three primary forms of data include historical data from past projects, planning data, and actual performance data, as follows.

- **Historical Data**. Most actual performance data originates with the developer. This data includes information collected by the software customer from past projects, as well as data collected by the developer from previous projects. This data is used to generate estimates and determine the feasibility of plans.

- **Plan Data**. The customer often produces initial planning data, which typically contains the budgets and schedules against which progress and expenditures are compared. Data must be collected from both initial plans and later re-planning and include incremental changes to plans. As the project evolves, the corresponding actual data on problems, progress, size, and effort will become available.

- **Actual Performance Data**. Many sources of data exist within the software development process. The number of software problems can be obtained from CM databases, grouped by severity, if they are properly structured. The number of hours expended, by activity, can be obtained from financial management records. Progress data usually comes from he detailed work plans maintained by technical managers and team leaders. Consistent use of project management tools facilitates data collection.

The number of SUs, lines of code, and changes to software and documents can usually be obtained from CM records and reports. Alternatively, a source code analyzer may be used. Product information, such as number of lines of code or pages, can also be captured during reviews and inspections. Note that in all these cases, the most efficient method of collecting the desired data depends on the nature of the software development process. Table 4.2 shows some typical sources of data.

Measurement Category	Electronic Source	Paper Source
Milestone Performance	Project Management System	Schedule
Personnel	Cost Accounting System Time Reporting System Estimation Tools	Time Sheets
Product Size and Stability	Static Analysis Systems CM System	Product Listing
Functional Size and Stability	Function Point Counting Systems CM System	Requirements Specifications
Defects	Problem Tracking System Test Automation System CM System Computer Aided Software Engineering (CASE) Tools/Interactive Development Environment (IDE) Tools Test Automation Tools	SPRs Review/Inspection Reports
Complexity	Static Analysis Systems	Review/Inspection Reports

Table 4.2 Typical Sources of Data

For important issues, look for sources of data that are available early. For example, if quality is a major concern try to identify sources of inspection data during design, rather than waiting for SPR data from testing.

Specify Measurement Implementation Requirements

The last task in the activity of integrating measurement into the software development process is to define data and implementation requirements for each selected measure. As much as possible, take advantage of the existing measurement opportunities defined in the preceding task.

This general specification guidance outlines the requirements related to defining and collecting measurement data. These requirements help define the overall measurement implementation

approach on the project and help convey to the developer how the measurement plan is to be implemented. The general requirements include the following:

- **Data Types**. Measurement data that represents plans, changes to plans, and actual information for each measure should be collected and reported. The developer should updated plans and estimates regularly. Effective insight can be derived early in the project by analyzing how the planning data is changing. Extremely stable plans may indicate that the developer is not adjusting to actual project events. For many projects, some plans and estimates are difficult to collect due to limitations of the software development process. For example, not everyone can adequately project the number of expected SPRs. In these cases, trends based on the periodic collection of actual data may be adequate to support measurement analysis requirements.

- **Measurement Definitions**. During this task, the developer identifies the actual measurement definitions to be used for each specified measure. These definitions sometimes vary over the course of the project, as software development processes are modified and updated. Changes to the definition and interpretation of any measure should be defined by the developer and relayed to the PMs. In many cases, this information is included in the periodic delivery of the measurement data. For many measures, such as lines of code, estimation methodologies and how the actual lines of code are counted may be different. This can result in variances between plans and actual data relate to definitions, not performance. These estimation inconsistencies should be identified. Many measures require that both estimation and actual counting methodologies be defined, as well as the "exit" criteria for measuring actual data. Definition of the measures is extremely important; it provides the basis for correct interpretation of associated data.

- **Data Dates.** For each measure, both the date that the measurement data was collected and the date that it is reported should be identified. This allows the timeliness of the data to be assessed and supports the correlation of related measurement data during analysis. For example in productivity calculation, the time period during which the number of lines of code is produced should correspond to the time period of the labor hours used to produce them. The time between data collection and delivery to the PM office should be minimized. This allows for timely analysis and feedback on the issues.

- **Collection Frequency**. Measurement data should be collected periodically, not by event. This is monthly on most projects; however, the period can be adjusted. Data may be collected more frequently as a milestone nears. For example problem report data is often collected and reported on a weekly basis during SI&T phases. The frequency of collecting and reporting measurement data should be consistent with the level of risk associated with each issue.

- **Measurement Scope**. If more than one organization is involved in developing the software for a project, measurement data should be collected from each organization

and identified by source. This is usually the case when one or more software subcontractors work under a prime contractor. In many instances, individual organizations have different software development processes, which result in different definitions for the same measure. This prevents the combination and aggregation of some types of measurement data from the different organizations. In these cases, the data from each organization must be managed and analyzed separately. For example, a system level productivity calculation may be invalid if subcontractors count labor hours and software size using different definitions. In some cases, different measures will be used by different organizations to address similar issues.

- **Project Phase**. Measures selected and integrated into the project should be applied to all life-cycle phases, including project planning, development, and sustainment engineering. For most measures, planning data will be available initially, followed by actual data as the project progresses and planned system process activities are implemented. Even when actual data is available, the related measurement plans and estimates should be updated periodically.

- **Data Reporting Mechanisms**. Reporting mechanisms for delivery of data from the developer may vary based upon the actual measures selected and internal software and project management processes of both the developer and the project office. Data for many measures, such as SPRs, are usually available from an existing CM database that can be accessed on a real-time basis. In other cases, such as with level of effort, size data, and schedule measures, the data can be formatted into electronic media and delivered. Some data may need to be delivered in hard-copy format. During tailoring, the developer identifies available mechanisms. The preferred method is electronic transfer of data on a periodic basis.

Project Measurement Plan

The results of the final activity of the tailoring process are documented in a Project Management Plan. The plan lists the issues and the measures required to address the issues. The plan describes the process used to collect and analyze the data. It explains how the developer and PMs use measurement results for decision making and communication within the project.

TPM Project Management Plan may be formal or informal. The plan should be modified, as required, to accommodate changes in information needs and software development processes. The plan may be produced as a separate document, or included in the Software Development Plan (SDP), the Software Maintenance Plan (SMP), or similar planning document. Regardless of the formality of the measurement plan, it should incorporate the following information:

- **Issues and Measures**. Lists identified issues and selected measures and show their relationships.

- **Data Elements**. Defines structures, attributes, and data items required for all measures.

- **Data Definitions**. Provides a complete and unambiguous definition for each data item. Defines methodologies used to calculate derived measures, such as productivity.

- **Data Sources**. Identifies specific sources, including person, tool, report, and activity, for all data items.

- **Level of Measurement**. Determines the level of detail for data items to be collected and delivered for analysis.

- **Aggregation Structure**. Defines structures for combining data items to provide system and other aggregations.

- **Frequency of Collection**. Specifies the frequency of data collection and delivery for analysis. This is typically monthly.

- **Method of Delivery**. Defines methods for accessing data, such as access to a database or electronic media.

- **Communication and Interfaces**. Identify POCs for all data sources, reports, and requests for clarification.

- **Frequency of Analysis and Reporting**. Determine how often measurement results will be provided to the project. This is typically monthly.

Table 4.3 contains a more detailed sample outline for a Project Management Plan.

Project Measurement Plan Outline

Part 1. Introduction
- Purpose
- Scope

Part 2. Project Description
- System Technical Characteristics
- Project Management Characteristics

Part 3. Measurement Approach

- How to Integrate Measurement Into the System Technical and Management Processes
- How to Collect and Use Data
- Measurement POCs (developer, subcontractors)
- Measurement Responsibilities
- Organizational Communications and Interfaces

Part 4. Description of Project System Issues
- Prioritized List of System Issues and Objectives

Part 5. System Measures and Specifications
- Include for Each Selected Measure (for each developer, if different)
  a. Measure name
  b. Issue measure maps
  c. Data items
  d. Attributes
  e. Aggregation structures
  f. Collection level
  g. Criteria for counting measured values
  h. Data definitions
  i. Estimation methodology
  j. Collection and reporting mechanisms
  k. Source of data
  l. Collection and reporting frequency

Part 6. Project Aggregation Structures
- Component Aggregation Structure, such as CSCIs, units
- System Activity Aggregation Structure, such as Requirements Analysis, Design, Integration, Performance, and System Qualification Test Aggregation Structure

Table 4.3 Sample Outline for Project Measurement Plan

The Project Management Plan is coordinated with the Risk Management Plan and the Financial Performance Management Plan. All significant quantifiable risks should be reflected in the Project Management Plan. The Financial Performance Plan is based on objective information produced by the measurement process, rather than on subjective assessments of percent complete or remaining effort. For small projects, this information can be included in one Project Management Plan.

Organizational Measurement Plan

Most large projects will require the development of a unique Project Management Plan. However, some organizations can define a Project Management Plan that covers many projects. This implies that a common measurement set can be defined for the organization. A common measurement set makes sense only for projects that share the following characteristics:

- Similar software issues.
- Common software development processes (standards, practices).
- Stable technology (languages, tools, environment, configuration, etc.).
- Similar application domains.

Imposing a standard measurement set in situations where these conditions are not satisfied may burden individual projects with unnecessary measurement requirements, while missing important project issues that should be tracked.

A common data set or normalization scheme may be necessary for other types of analysis to support process improvement and business purposes. Although, this document focuses on project-level analysis, it provides a basis for organizational and executive-level measurement. Recording characteristics that drive decisions during the measurement selection activity may be important later to determine how to normalize data for these purposes.

In addition to project-specific measurement requirements discussed in this section, other users may have other valid requirements that the project's measurement process must address. Other users include executive managers performing an oversight function and software engineering process groups working on process improvement. Most of the data required by these other users originates at the project level. Obtaining good data for executive review and process improvement depends on establishing an effective project-level measurement process.

Most organizations require formal reporting of actual cost and schedule progress against budget baselines. The financial performance management system should be based on results from the measurement process.

Consider measurement requirements from all sources when developing a Project Measurement Plan. This enables a measurement analyst to minimize redundancy and inefficiency that can result from multiple data collection efforts. Focusing on measures and analyses that benefit multiple users maximizes the value of the implemented measurement process.

This page intentionally left blank

EFFECTIVE USE OF MEASUREMENT DATA

After TPM is planned and applied the results are presented to the project/task management. This illustrates various ways measures collected for TPM can be presented to and used by project/task management to identify, track, analyze, and resolve issues. The following examples of information presentation can be used a tool to communicate status and other additional information to all groups within the STO or the developer organization(s).

Tracking Test Effort Effectiveness

The effectiveness of the testing effort is measured constantly throughout the SI&T process. Measurements specific to the following three items are discussed in this section:

a. The overall test effort in relation to software requirements.

b. The progress of the test effort in relation to test requirements.

c. The effectiveness of SPR identification and resolution.

These measurements address schedule and progress issues. Additionally, the measurements give an indication of software project maturity and readiness for release.

Traceability

Software requirements are identified in the SRD and used as the basis for creating the STP. Test requirements are derived from software requirements and decompose to specific test cases in the STD. Software and test requirements are organized and controlled through CM procedures. One aspect of organizing and controlling requirements is traceability. Test requirements are traced to software requirements, and conversely, software requirements are traced to test requirements.

Test cases decompose to test case procedures. Test case procedures are traced to test requirements through test cases. As test case procedures are executed the Pass/Fail results are tracked using standard test execution processes.

Traceability provides the ability to measure the testing effort in terms of software requirements. Since the quantity of software requirements remains static from the beginning to the end of the SI&T process through the end, they can be used as a measurement of effort expended on SI&T.

Measuring Test Effort Relative To Software and Testing Requirements

Reports are generated at regular intervals to define the number of software requirements tested and the number of software requirements remaining to be tested. This information is presented in reports that tabulate software requirements that are:

d. Traced to test requirements associated with test procedures that were executed and passed.

a. Traced to test requirements associated with test procedures that were executed and failed.

b. Traced to test requirements that are associated with test procedures that have not been executed.

c. Traced to test requirements not associated with test case procedures.

d. Not traced to test requirements.

Table 5.1 depicts a sample of how these numbers are used to measure SI&T production through software requirements.

	Software Requirements	Planned	Actual
a.	Traced to test requirements associated with test case procedures which have been executed and passed	N/A	500
b.	Traced to test requirements that are associated with test case procedures which have been executed and failed	N/A	50
Total Tested	Total Software Requirements Tested (a + b)	500	550
c.	Traced to test requirements that are associated with test case procedures which have not been executed	100	75
d.	Traced to test requirements that are not associated with test case procedures	100	150
e.	Not traced to test requirements	100	25
Total SR	Total Software Requirements (a + b + c + d + e)	800	800
	Test Execution Remaining (c / (a + b)) %	16.67%	12.00%
	Test Development Remaining (d / Total SR) %	12.50%	3.13%
	Test Requirement Development Remaining (e / Total SR) %	12.50%	3.13%

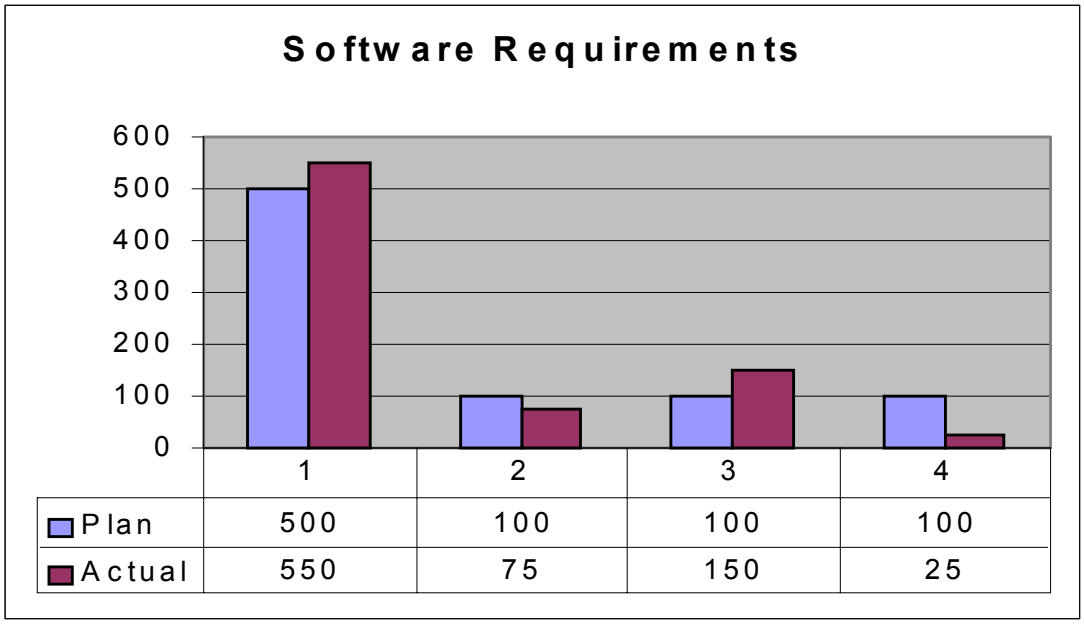
Table 5.1 Sample Measurement of SI&T Production, Planned Versus Actual

The total of (a.) plus (b.) describes the quantity of software requirements tested. The total indicated by adding (a.) plus (b.) can be compared to the quantity of software requirements projected to be completed, which will indicate the planned versus actual testing progress. The total software requirements tested can be compared to the quantity in (c.) to derive the percentage of the testing effort that remains.

The quantity in (d.) indicates the test case procedure development effort remaining. It can be used as a planning guide, as well as a comparison of planned versus actual test effort.

The quantity in (e.) indicates the effort remaining for defining test requirements and creating test case procedures.

Charts similar to Figure 5.1 are used to display the relationship of actual versus planned test effort, based on software requirements.



	1	2	3	4
Plan	500	100	100	100
Actual	550	75	150	25

1 Total number of software requirements tested.

2 Software requirements traced to test case procedures not tested.
3 Software requirements not associated with test case procedures.
4 Software requirements not traced to test requirements.

Figure 5.1 Software Requirements Testing, Planned Versus Actual

Figure 5.1 contains four sections, each with two columns. Sections are identified in the legend at the bottom of the chart. Section 1 indicates the quantity of software requirements tested. Section 2 indicates the quantity of software requirements for which test case procedures are developed that are awaiting execution. Sections 1 and 2 indicate test execution progress relative to the plan in a positive relationship. If the Actual column is greater than or equal to the Planned column, the effort is considered successful. Beyond detailing planned versus actual effort, Section 2 can be matched to Section 1. The relationship of Section 2 to Section 1 is a scheduling aid for the test manager.

Section 3 and Section 4 indicate the progress of test development relative to the plan in a negative relationship. If the Actual column is greater than the Planned column, the effort is not considered successful. Section 3 indicates the quantity of software requirements for which test cases are developed but test case procedures are not. Section 4 indicates the quantity of software requirements for which test cases are not developed.

As the SI&T effort progresses, columns will grow higher from right to left on the chart. Section 4 will have the highest numbers at the beginning of the project, and Section 1 will have the highest numbers at the end of the project. At given points in the process, Sections 4, then Section 3, and finally Section 2, will be empty.

Measuring the Effectiveness of SPR Identification and Resolution

The primary purpose of any testing program is the delivery of error free software that will be integrated to become a system that satisfies defined design requirements. To achieve the goal of error free software identification, tracking, and resolution of problems discovered during software testing is a primary objective during the testing phases. Identification, tracking, and resolution of problems during the testing phases are accomplished using SPR forms. The SPRs are organized, controlled, and counted to indicate software readiness for release.

While the total number of SPRs is useful in planning efforts, it is not always a good indication of the quality or effectiveness of independent testing. High quality SU testing, performed by the software developer, will lower the number of problems detected and reported by the test group.

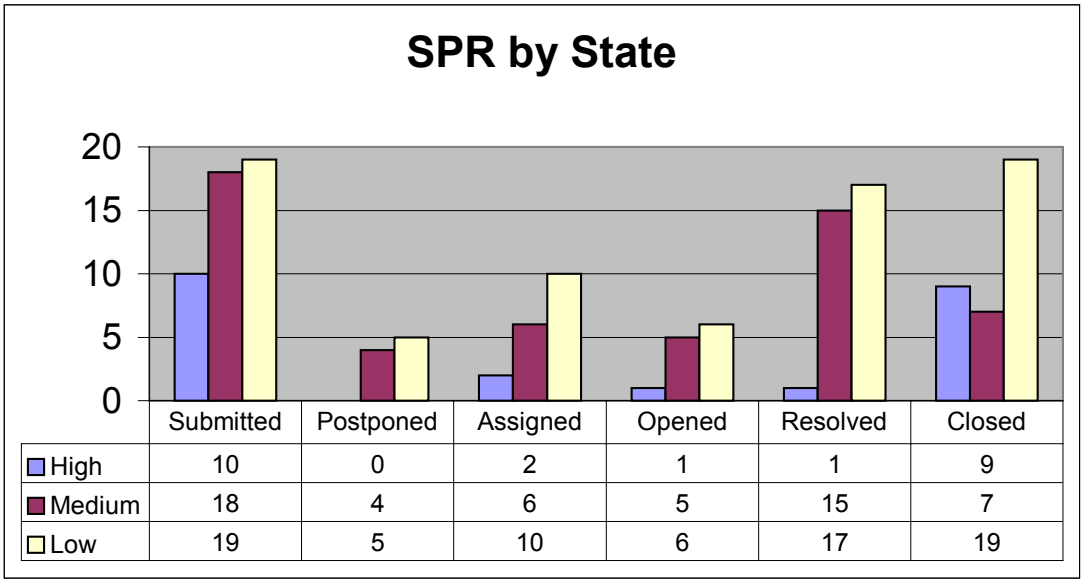
Useful measurements in reports designed to report problems identified fall into three categories. The first is the number of SPRs in each state of the SPR life cycle, at a particular point in the testing process. The second category is the average time for SPRs to transition from one state to the next, in normal progression. The third category is the number of SPRs that do not follow a normal progression. Each category is discussed separately later in this document.

Regardless of the category, SPR measurement is accomplished by providing a sub-measurement by level of severity.

System Problem Reports By State of System Problem Report Life Cycle

Reporting by SPR life cycle state shows the number of SPRs a) submitted, b) postponed, c) assigned, d) opened, e) resolved, and f) closed. Each of these numbers is important for planning and scheduling. A separate total of SPRs in the duplicate state is generated to reflect the entire SPR life cycle. A high number of SPRs reported, minus duplicates, may indicate ineffective SU testing. A high number of postponed SPRs may indicate ineffective planning, design, or development efforts. A high number of duplicate SPRs may indicate ineffective integration testing (duplication of effort) or a lack of training for test engineers, in the software application being tested.

The sample chart in Figure 5.2 contains six sections, one for each state of the SPR life cycle of non-duplicate SPRs. The Submitted section indicates SPRs that have not gone through an SPR Review. The Postponed section depicts reviewed SPRs that are awaiting future action. The Assigned section depicts reviewed SPRs that are not been scheduled for repair by the system development group. The Opened section depicts SPRs scheduled for repair by the system development group. The Resolved section depicts SPRs repaired by the system development group that are awaiting re-test by the test group. The Closed section depicts successfully tested repaired SPRs.



	Submitted	Postponed	Assigned	Opened	Resolved	Closed
High	10	0	2	1	1	9
Medium	18	4	6	5	15	7
Low	19	5	10	6	17	19

Figure 5.2 System Problem Report Status by States

Each section in Figure 5.2 contains three columns that indicate levels of severity. The number of SPRs in each section, by level of severity, can be used as a planning tool to gauge the effort needed from the system software group and the test group to resolve outstanding problems.

The chart in Figure 5.3 contains three columns, High, Medium, and Low. Each column describes a level of severity. Each column indicates the total number of SPRs and the number of duplicate SPRs. Duplicate SPRs are determined during the SPR Review process. Specifically, these are duplicate SPRs not closed at the time of the review. A high percentage of duplicates may indicate duplication of effort by individual test engineers in the test group. For reporting purposes, SPRs awaiting review, postponed during a review, or passed on to the system development group are shown as one total. The total number of SPRs, by level of severity, depicted in Figure 5.2 are shown in the lower portion of the columns of Figure5.3 as the Total. The number of duplicate SPRs is compared to the Total, by degree of severity, and shown in the upper portion of Figure 5.3.

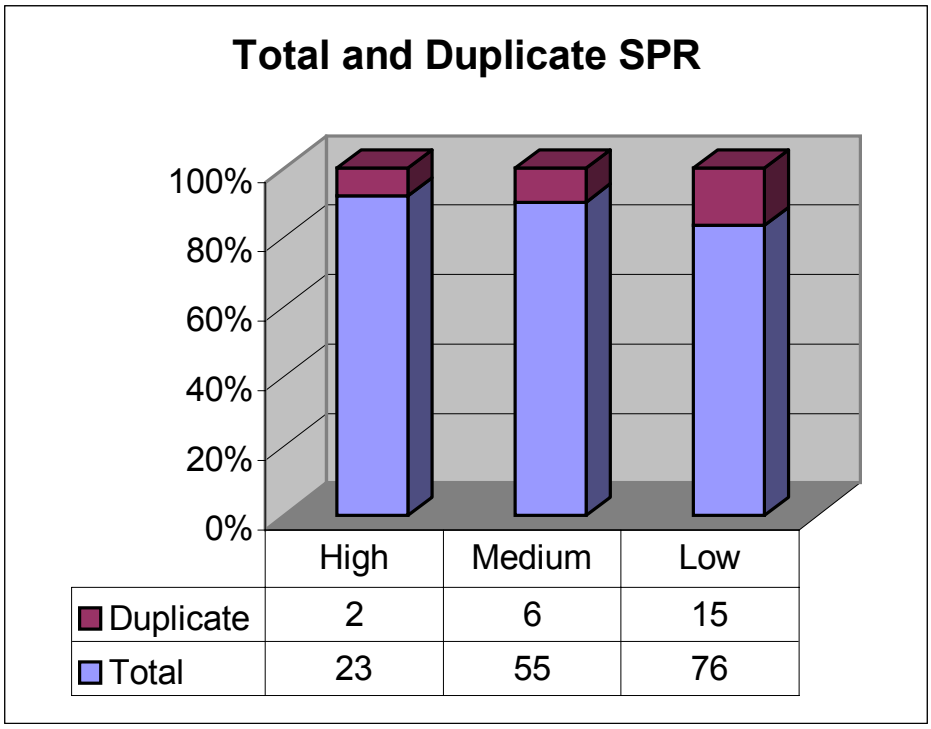


Figure 5.3 Total Versus Duplicate System Problem Reports

Duration Between States of System Problem Report Life Cycle

The time between states, as SPRs progresses through key milestones in the life cycle, is an important indicator of the effectiveness of the SI&T process. In normal progression, SPRs move from submitted through assigned, opened, resolved, and then closed. A significant time lapse between submitted and assigned may indicate an ineffective SPR Review process. A significant time lapse in moving an SPR from assigned to opened, or from opened to resolved, may indicate workflow problems within the software development group. The time lapse in transitioning from resolved to closed might indicate workflow problems in test group efforts.

SPR milestones include assigned, resolved, and closed. These milestones indicate responsibility for action passing between the test group and the system development group. A sample chart depicting SPR progression is provided in Figure 5.4.

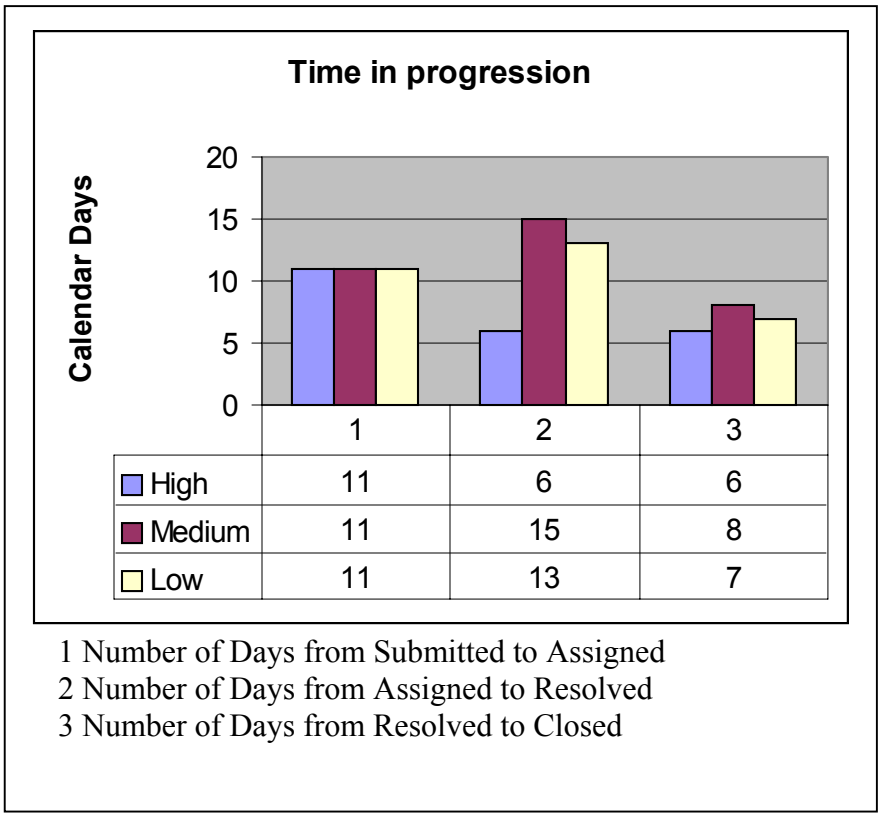


Figure 5.4 System Problem Reports Progression Times

Figure5.4 contains three sections, each with three columns. Sections are identified in the legend at the bottom of the chart. Section 1 shows the average number of calendar days before an SPR is reviewed. This is an indicator of the efficiency of the SPR Review process. Section 2 shows the average number of days the SD group requires to repair problems. The SD group is responsible for the repair once the SPR is assigned through the SPR Review process. After the problem is repaired, the Test group schedules the re-test. Re-tests are conducted after the SD group declares the SPR resolved. Section 3 shows the average number of days it takes the Test group to conclude the re-test or re-open the SPR, if the re-test fails.

The columns in each section depict levels of severity; High, Medium, and Low.

Only SPRs that follow a normal progression are included in a report, such as depicted in Figure5.4. SPRs that should not have been submitted and do not follow normal progression are the topic of sub-section, 5.2.2.

Abnormal Progression

SPRs closed from any state other than resolved is an indication of an ineffective SI&T process. Moving a SPR from submitted to closed, after the SPR review indicates a lack of test engineer training. Moving from assigned to duplicate or from assigned to closed indicates a problem in

the SPR Review process. Abnormal progression is reported to assist in identifying problem areas and improving the process.

The chart in Figure 5.5 is an example of a chart depicting SPRs closed without being resolved

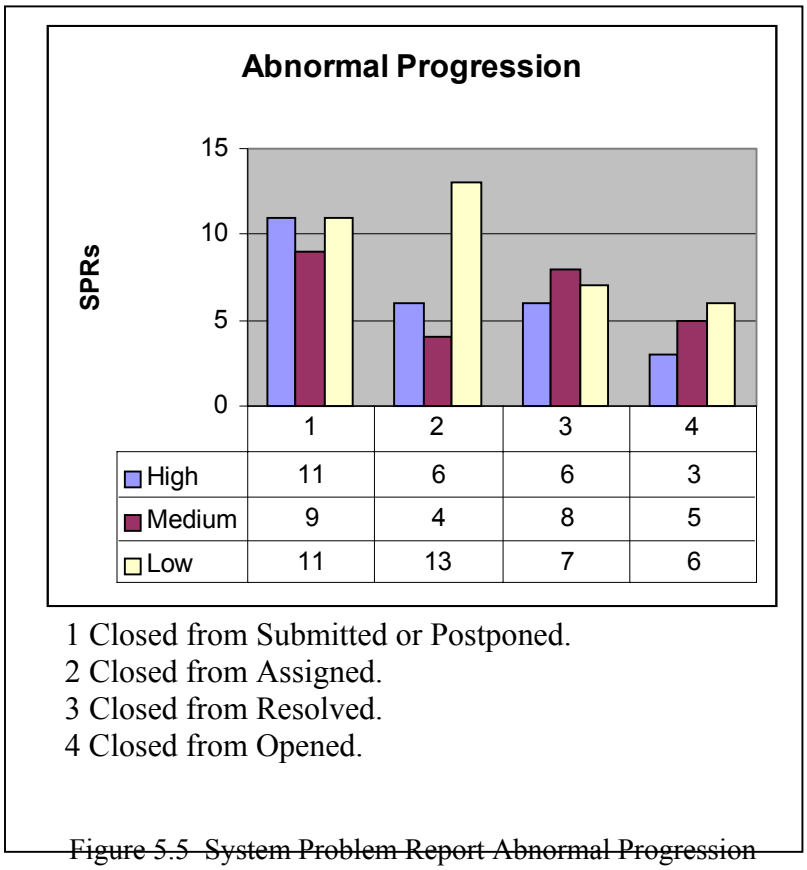


Figure 5.5 System Problem Report Abnormal Progression

The chart in Figure 5.5 contains four sections, each with three columns. Sections are identified in the legend at the bottom of the chart. Section1 depicts the number of SPRs detected as incorrect during the SPR Review process. A significant number in Section 1 may indicate a lack of test engineer training specific to the software application being tested. Sections 2, 3, and 4 depict the number of SPRs not detected as erroneous during the initial SPR Review process. Significant numbers in these sections may indicate a problem with user documentation, system documentation test case development, the SPR Review process, or any combination of the three areas.

This page intentionally left blank

GLOSSARY

Aggregate

A mass of distinct things gathered into a total or whole.

Aggregation Level

Effective measurement analysis and reporting requires that the data be aggregated to higher levels of the of the software components and project organizational structure. The aggregation levels define the different ways the measurement data can be grouped and organized for reporting on the project. The aggregation levels describe how the measurement data relates to an existing product and process structures. The organization that allows the measurement results to be combined, and later decomposed, into meaningful pieces of information.

Aggregation Structure

The structure used to define the data according to the defined aggregation levels. The levels may describe the personnel and management structure of the project, or the configuration of physical components of the project. All entries in a structure should be of the same type, such as software modules. However, these entries may reside at various levels of the structure, such as software modules at the unit level, CSCI, or integrated level of the software architecture.

Application

(1) A complete, self-contained program that performs specific function(s) directly for the user.

(2) In the TPM process this term refers to one of the two basic measurement activities which comprise the system measurement process. The application activity involves collecting, analyzing, and acting upon the measurement data.
See **Tailoring**

Automated Test Script

A computer readable set of instructions that performs a sequence of steps, sub-steps, or other actions, performed serially, in parallel, or in some combination of consecution, that creates the desired test conditions that the test case is deigned to evaluate.

Baseline

A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

Baseline Control

Baseline control is the process that regulates approved and released versions of all software, documentation, and the test environment throughout the test life cycle.

Black Box Testing

This is testing associated with functional testing where the object being tested is treated as a black box. In this type of testing the test object is subjected to inputs and outputs that are verified for conformance to prescribed specifications.

Capacity Testing

Attempts to simulate expected customer peak load operations in order to ensure that the system performance requirements are met. It does not necessarily exercise all of the functional areas of the system, but selects a subset that is easy to replicate in volume. It will ensure that functions which are expected to use the most system resources are adequately represented.

Change Control

The process by which problems and changes to the software, documentation, and test environment are evaluated, approved, rejected, scheduled, and tracked.

Computer Aided Software Engineering (CASE)

A technique for using computers to help with one or more phases of the software life cycle, including the systematic analysis, design, implementation and maintenance of software. Adopting the CASE approach to building and maintaining systems involves software tools and training for the developers who will use them.

Computer Software Configuration Item (CSCI)

An aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

Configuration Control

An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.

Configuration Item (CI)

Hardware or software, or an aggregate of both, which is designated by the project configuration manager (or contracting agency) for configuration management.

Configuration Management (CM)

A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

Configuration Management Office (CMO)

The Configuration Management Office (CMO) is the officiator of the project life cycle CM process.

Criteria

A standard, rules, or tests by which something can be judged.

Critical Defect

See Criticality

Criticality

The assessment of the impact upon a system of a given error, defect, problem, or discrepancy during the life cycle of a system.

The definition of critical and non-critical system defects or problems should be addressed at a management level and can be different for each system. For any given system error, defect, problem, or discrepancy, an appropriate impact value (i.e., priority) will be assigned.

An example of impact values with the corresponding priority numbers is presented below as contained in IEEE/EIA Std-12207, 1998. The priority that will apply if a problem can result in one or more of these impacts:

PRIORITY IMPACT

(3) a.) Prevent the accomplishment of an operational or mission essential capability.

b.) Jeopardize safety.

c.) Cause significant technical, cost, or schedule risks to the project or to life cycle support of the system.

(4) a.) Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is known.

b.) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, and no work-around is known.

(5) a.) Adversely affect the accomplishment of an operational or mission essential capability, but a work-around solution is known.

b.) Adversely affect technical, cost, or schedule risks to the project or to life cycle support of the system, but a work-around is known.

(6) a.) Results in user/operator inconvenience or annoyance, but does not affect a required operational or mission essential capability.

b.) Results in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of the responsibilities of these personnel.

(7) a.) This priority denotes any other effect.

Customer

The organization that procures software systems for itself or another organization.

Developer

An organization that develops software products. The term "develop" may include develop, modification, integration, reengineering, sustaining engineering, maintenance, or any other activity that results in software products. The developer may be a contractor or a government agency.

Discrepancy

An inconsistency or disagreement found during testing between the actual and expected test results.

Document

A data medium and the data recorded on it that generally has permanence and can be read by a human operator or machine. Often used to describe human readable items only (e.g., technical documents, design documents, requirements documents, etc.).

Documentation

(8) A collection of documents on a given subject.

(9) The management of documents, that includes the actions of identifying, acquiring, processing, storing, and disseminating.

(10) Any written or pictorial information describing, defining, specifying, reporting or certifying activities, requirements, procedures, or results.

Driver

A software program that exercises a system or system component by simulating the activity of a higher level component.

Emulation

One system is said to emulate another when it performs in exactly the same way, though perhaps not at the same speed. A typical example would be the emulation of one computer by (a program running on) another. You might use emulation, as a replacement for a system whereas you would use a simulation if you just wanted to analyze it and make predications about it.

Emulator

Hardware or software that performs emulation.

Entry Criteria

A set of decision making guidelines used to determine whether a system under test is ready to move into, or enter, a particular phase of testing. Entry criteria tend to become more rigorous as the test phases progress.

Environment

The infrastructure in which a system is executing, consisting of hardware, operating system software, interfaces, etc.

Exit criteria

A set of decision-making guidelines used to determine whether a system under test is ready to exit a particular phase of testing. When exit criteria are met, either the system under test moves on to the next test phase or the test project is considered complete. Exit criteria tend to become more rigorous as the test phases progress.

Final System Test Report (FSTR)

Used to determine whether system testing is completed and to assure that software is ready for production.

Hardware Configuration Item (HWCI)

An aggregation of hardware that is designated for configuration management and treated as a single entity in the configuration management process.

Independent Verification and Validation (IV&V)

The verification and validation of a software product by an organization that is both technically and managerially separate from the organization responsible for developing the product.

Indicator

A measure or combination of measures that provides insight into a system issue or concept. TPM frequently uses indicators that are comparisons, such as planned versus actual measures. Indicators are generally presented as graphs or tables.

Integration

Combining software or hardware components or both into an overall system.

Integration Testing

The period of time in the software lifecycle during which the application is tested in a simulated production environment to validate the communications and technical architecture of the system. This test phase occurs when all the constituent components of the system under test are being integrated.

Interactive Development Environment (IDE)

A system for supporting the process of writing software. Such a system may include a syntax-directed editor, graphical tools for program entry, and integrated support for compiling and running the program and relating compilation errors back to the source code.

Interface

(11) A shared boundary (e.g., a hardware component linking two devices or registers, or a portion of storage accessed and/or modified by two or more computer programs).

(12) To interact or communicate with another system component.

Interface Requirement

A requirement that specifies a hardware, software, or database element with which a system or system component must interface, or that sets forth constraints caused by such an interface.

Interface Specification

A specification that sets forth the interface requirements for a system or system component (e.g., the software interface specification document).

Interface Testing

Tests conducted to ensure that program or system components correctly pass data and/or control to one another.

Issue

An area of concern where obstacles to achieving program objectives might arise. Issues include risks, problems, and lack of information. These three types of issues are defined as:

(13) Risk -- An area of concern that could occur, but is not certain. A risk is a potential problem. Risks represent the potential for the realization of unwanted, negative consequences from a project event. For example, a project plan may be based on the assumption that a COTS component will be available on a given date. There is a possibility (probability) that the COTS may be delayed and have some amount of negative impact on the project.

(14) Problem -- An area of concern that a project is currently experiencing or is relatively certain to experience. For example, a shortage of staff with the right skills may be an actual problem that is delaying the project.

(15) Lack of Information -- An area where the available information is inadequate to reliably predict project impact. Thus, satisfaction of project objectives is questionable even if no problems or risks are present. For example, lack of information about the size of the software to be developed could result in the project "discovering" that it has more work to do than originally planned.

Measure

The result of counting or otherwise quantifying characteristics of a process or product. Measures are numerical values assigned to system attributes according to defined criteria.

Measured (or actual) Value

Actual, current measurement data, such as hours of effort expended or line of code produced.

Measurement

The process of assigning quantitative values of system properties, according to some defined criteria. This process can be based on estimation or direct measurement. Estimation defines planned or expected measures. Direct measurement results in actual measures.

Measurement Analysis

The uses of measurement data to identify problems, assess problem impact, project an outcome, or evaluate alternatives related to system issues.

Measurement Analyst

The person(s) or team responsible for tailoring and applying system measures for a given project or task.

Measurement Information

Knowledge derived from analysis of measurement data and measurement indicators.

Milestone

A scheduled event for which some project or task member or manager is held accountable. A milestone is often used to measure progress.

Module

A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading.

Note: *The terms 'module', 'component', and 'unit' are often used interchangeably or defined to be sub-elements of one another in different ways depending on the context.*

Non-Critical Defect

See Criticality

Performance Testing

The period of time in the system or software development lifecycle during which the response times for the application are validated to be acceptable. The tests ensure that the system environment will support production volumes, both batch and on-line.

Priority

A measure of the elements of importance related to the repair of a system problem that are not considered in defining the severity of a system problem.

Project Manager (PM)

The official responsible for acquiring, developing, or supporting a system to meet technical, cost, schedule, and quality requirements. Acquisition, development, and support will include both internal tasks and work that is contracted to another source.

Quality Assurance (QA)

A planned and systematic pattern of all actions necessary to provide adequate confidence that the product optimally fulfils customers expectations.

Quality Control (QC)

The assessment of product compliance. Independently finding deficiencies assures compliance of the product with stated requirements.

Requirement

(16) A condition or capability needed to solve a problem or achieve an objective.

(17) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. The set of all requirements forms the basis of development.

Regression testing

Part of the test phase of software development where, as new modules are integrated into the system and the added functionality is tested, previously tested functionality is re-tested to assure that no new module has corrupted the system.

Risk

An area of concern that may occur, but is not certain. A risk is a potential problem. Risks represent the potential for the realization of unwanted, negative consequences from a project event. For example, a project plan may be based on the assumption that a commercial off the shelf (COTS) component will be available on a given date. There is a possibility (probability) that the COTS may be delayed and have some amount of negative impact on the project.

Severity

The degree to which a problem adversely influences the system's operation or the overall test effort.

Simulation

Attempting to predict aspects of the behavior of a system by creating an approximate (mathematical) model of it. This can be done by physical modeling, by writing a special-purpose computer program or using a more general simulation package, aimed at a particular kind of simulation. Typical examples are aircraft simulators or electronic circuit simulators.

Simulator

Hardware or software that performs simulation.

Software Design Specification (SDS)

A document that records the design of a system or system component; typical contents include: system and/or component algorithms, control logic, data structures, data set use, input/output formats, and interface descriptions.

Software Development File (SDF)

The developer shall document the development of each Computer Software Unit (CSU), Computer Software Component (CSC), and CSCI in Software Development Files (SDF). The developer shall establish a separate SDF for each CSU or a logically related group of CSUs, for each CSC or a logically related group of CSCs, and for each CSCI. The developer shall document and implement procedures to establish and maintain SDFs. SDFs may be generated, maintained, and controlled by automated means. To reduce duplication, SDFs should not contain information provided in other documents or SDFs. The set of SDFs shall include (directly or by reference) the following information:

(18) Design considerations and constraints.

(19) Design documentation and data.

(20) Scheduling and status information.

(21) Test requirements and responsibilities.

(22) Test case, test case procedures, and results.

Software Life Cycle

The phases a software product goes through between when it is conceived and when it is no longer available for use. The software life cycle typically includes the following: requirements, analysis, design, construction, testing (validation), installation, operation, maintenance, and retirement. The development process tends to run iteratively through these phases rather than linearly; several models (spirals, waterfall, etc.) have been proposed to describe this process. Other processes associated with a software product are: quality assurance, marketing, sales, and support.

Software Management Plan

A project plan for the development of the software component of a system or for the development of a software product.

Software Requirements Document (SRD)

This is a formal document derived from the Software Requirements Specification (SRS) that sets forth the requirements, specifications, and standards for a system (e.g., a software product). Typically included are functional specifications and requirements, performance specifications and requirements, interface specifications and requirements, design specifications and requirements, and development requirements and standards.

Software Requirements Specification (SRS)

A specification that sets forth the requirements for a system component; (e.g., a software product). Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards.

Software Tool

Computer programs used to help develop, test, analyze, or maintain another computer program or its documentation.

Specification

Documentation containing a precise, detailed, verifiable description of particulars with respect to the requirements, design, function, behavior, construction, or other characteristics of a system or system component.

Stub

(23) A dummy procedure used when linking a program with a run-time library. The stub routine need not contain any code and is only present to prevent "undefined label" errors at link time.

(24) A local procedure in a remote procedure call (RPC). The client calls the stub to perform some task and need not necessarily be aware that RPC is involved. The stub transmits parameters over the network to the server and returns the results to the client/caller.

System

- Any large program.
- The entire computer system, including the input/output devices, supervisor program or operating system and possibly other software.

System Problem Report (SPR)

A form that is used to record a discrepancy discovered during the Integration Test, Performance Test and System Qualification Test phases of the SI&T process concerning a Computer Software Configuration Item, a software system or subsystem, other software related items, and associated documentation.

System Problem Report (SPR) Status Report

The System Problem Report Status Report is used during the SPR Status Review to determine if the SPRs are being processed appropriately and expeditiously.

System Testing

The period of time in the software lifecycle during which the implementation of each requirement is validated.

Tailoring

In the TPM process, this term refers to one of the two basic measurement activities, which comprise the system measurement process. The tailoring activity includes identification and prioritization of program issues, selection and specification of appropriate system measures, and integration of the measurement requirements to the developer's system process.
See **Application**.

Test

The process of exercising a product to identify differences between expected and actual behavior.

Test Artifacts

An item created during the system integration and test process that is preserved upon completion of the test process (e.g., test plans, requirements documentation, automated test scripts, and test documentation).

Test Case

A description of a test to be executed for or focused on a specific test aim.

Test Case Procedures

A sequence of steps, sub-steps, and other actions, performed serially, in parallel, or in some combination of consecution, that creates the desired test conditions that the test case is designed to evaluate.

Test Case (Setup) Suite

The steps required to configure the test environment for execution of a test case.

Testing Condition

System state or circumstance created by proceeding through some combination of steps, sub-steps, or actions in a test case.

Testing Environment

The infrastructure in which the test is performed, consisting of hardware, system software, test tools, and procedures.

Test Plan

In a test plan the general structure and the strategic choices with respect to the test to be executed are formulated. The test plan forms the scope of reference during execution of the test and also serves as an instrument to communicate with the customer of the test. The test plan is a description of the test project, including a description of the activities and planning, therefore it is *not* a description of the tests themselves.

Test Readiness Review (TRR)

Review conducted to determine whether a software test phase has been completed and to assure that the software is prepared for the next step in the formal integration and testing procedures. Software test procedures and results are evaluated, for compliance with the software testing requirements and system descriptions, for adequacy in accomplishing testing goals. Also, provides the forum for updating and revising operational and supporting documentation.

Test Resources

Aids that are used by a test tool for collecting, tracking and controlling information. This information is:

(25) Software requirements defined in the Software Requirements Document.

(26) Test requirements defined in the System Test Description.

(27) Automated test case scripts as defined in the System Test Description.

(28) SPRs as determined at each phase of the System Integration and Testing process.

This information is controlled by Configuration Management at the end of the SI&T process for use whenever further testing may be conducted, using a testing tool, during the remaining lifecycle of the software or system.

Test Tools

The software, hardware, systems, or other instruments that are used to measure and test an item.

Traceability

Degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor, successor, or master-subordinate relationship to one another (e.g., the degree to which the requirements and design of a given software component match).

Unit

The lowest element of a software hierarchy that contains one or more of the following characteristics:

(29) A unit comprising one or more logical functional entities.

(30) An element specified in the design of a computer software component that is separately testable.

(31) The lowest level to which software requirements can be traced.

(32) The design and coding of any unit can be accomplished by a single individual within the assigned schedule.

Unit Test

The process of ensuring that the unit executes as intended. This usually involves testing all statements and branch possibilities.

Version

One of a sequence of copies of a system, each incorporates new modifications.

Version Identifier

A unique identifier assigned to baseline software, documentation, and test environment.

Version Control

The process by which all changes to the software, documentation, and test environment are compiled and built into a new version of the system.

Version Control Report

A report that details all changes and enhancements made to current version of the software, documentation, and test environment.

White Box Testing

This type of testing is associated with structural testing in which the testing can be characterized as being tied to implementation details, such as control methods, database design, coding details, and logic paths. The process of how an individual input is treated to produce a given output is ascertained. Structural testing is sometimes referred to as "clear box testing" since white boxes are considered opaque and do not really permit visibility into the code.

Work Breakdown Structure (WBS)

A work breakdown structure for software defines the software-related elements associated with program work, work activities, and products. Many measures are aggregated and analyzed at various WBS levels.

This page left intentionally left blank

BIBLIOGRAPHY

Black, Rex. *Managing The Testing Process,* Redman, WA: Microsoft Press, 1999

Koomen, Tim and Pol, Martin. *Test Process Improvement*, Essex, England, UK: Pearson Education Limited, 1999

Carnegie Mellon University, Software Engineering Institute. The Capability Maturity Model: Guidelines for Improving the Software Process, 1995.

Institute of Electrical and Electronics Engineers (IEEE). "Glossary of Software Engineering Terminology", IEEE-Std-610.12, 1990.

Institute of Electrical and Electronics Engineers (IEEE)/Electronic Industries Alliance (EIA). "Software Life Cycle Processes", IEEE/EIA Std-12207, 1998.

U.S. Department of Education. "SFA System Integration & Testing Approach, SFA Modernization", Undated.

U.S. Department of Education. "SFA Enterprise Configuration Management Approach, SFA Modernization", Undated.

Federal Systems Integration and Management Center (FEDSIM). "FEDSIM Writers Guide, Version 2", May 1994

Practical Software Measurement: A Foundation for Objective Project Management, Version 3.1, 17 April 1998, Office of the Under Secretary of Defense for Acquisition and Technology, Joint Logistics Commanders Joint Group on Systems Engineering

Free On-Line Dictionary Of Computing web site at www.foldoc.org

This page intentionally left blank

APPENDIX A

MEASUREMENT TAILORING PLAN

This page left intentionally left blank

A1. Measurement Tailoring Example

This appendix describes the TPM tailoring process as it is applied to a project. An example of a fictional project scenario is provided to demonstrate use of the TPM process to select a set of software measures.

A1.1 Project Scenario

During the project-planning phase of a large, real-time sensor system software upgrade, the project office learned that the updated system would be deployed earlier than originally planned. The planning efforts completed to date had identified some significant constraints with respect to schedule, and this change increased the schedule risk. The PM decided to implement a measurement process to help guide the project through these challenges.

First, the key characteristics of the project were identified and documented. This information is summarized as follows:

- Large, real-time sensor system
- Existing system baseline
- Approximately 1.5 Million lines of source code to be implemented
- Multiple software languages (Ada, C+, and Assembly)
- Multiple developers working under a prime contractor responsible for system integration
- Average software development process maturity across all organizations
- Funding limits

Due to the schedule risk and the large amount of functionality to be implemented in a short time, the project office required that the developer use COTS software components. The developer was instructed to reuse a considerable amount of legacy software, adopt an open-systems architecture, and apply commercial software development process standards.

A1.1.1 Identify And Prioritize Project Issues

Following TPM selection and specification approach, a planning workshop was held to identify and prioritize project-specific software issues and then select appropriate measures for those issues. This workshop was a facilitated session with representatives from the project management team, the developer, and subcontractors. After the workshop, a subset of participants was designated to form an IPT to develop and implement a measurement plan based on the workshop results.

During the workshop, participants developed a list of issues affecting the project. The issues included risks identified through the formal risk management process, the project objectives and constraints specified in the contract, and issues identified based on experience from previous projects. This activity produced a list of areas of concern for the project. These issues are consolidated into a set of prioritized, project-specific issues with related sub-issues as outlined in Table A.1.

Issue/Sub-Issue	Priority
Are schedule milestones met? • Is integration and test progress adequate to meet the delivery date? • Do incremental builds contain the specified functionality?	1
Is the productivity rate sufficient to meet plans? • Were size estimates used for cost and schedule plans correct? • Will the planned COTS/reuse meet allocated requirements or will new code be required?	2
Are there sufficient resources to complete the development?	3
Are requirement changes impacting the development?	4
Does the project meet quality requirements, as measured by the number of SPRs?	5

Table A.1 Issues and Priorities

The primary risk to the project is the short development schedule. The project had originally been "sold" on new capabilities and the use of advanced technologies. Using advanced technologies increased the overall technical risk of the software development. The need to deliver the system earlier than expected increased the concern.

A1.1.2 Select and Specify Project Measures

The issues were mapped to TPM common issues, as shown in Table A.2. TPM tables were reviewed to help determine the best measurement categories and associated measures to use to provide the required information. This selection also considered the availability of measures from the software development process. Figure A.2 lists the measures that resulted from the selection activity.

Project Specific Issue	TPM Common Issue	Categories	Measures
Schedule	Schedule and Progress	Milestone Performance	Milestone Dates
		Work Unit Progress	Component Status (SU, Integration, Performance, and SQT) Requirement Status
		Incremental Capability	Version Content-Function
Productivity	Development Performance	Productivity	Product Size/Effort Ratio
	Growth and Stability	Product Size and Stability	Lines of Code
	Technical Adequacy	Technology Impacts	Size by Origin
Resources	Resources and Cost	Personnel	Effort
Requirements	Growth and Stability	Functional Size and Stability	Requirements
Quality	Product Quality	Defects	SPRs

Table A.2 Measure Mapping

For purposes of this example, only the selection of the "Schedule and Progress" measures is discussed below. The categories of Milestone Performance, Work Unit Progress, and Incremental Capability were selected to address schedule. The Milestone Performance category was selected because this provided a high-level overview of schedule progress, and because Gantt charts were already being used to manage the project.

Work Unit Progress measures were selected to track development activities due to the amount of COTS and reused code to be implemented. The focus was on the selection of requirements-

oriented measures and measures that provide progress information for integration and test, rather than for design and implementation.

The build content measure was selected because the team determined that it was important to ensure that each of the incremental builds incorporated all planned functionality.

The team needed to be aware of any functionality deferment early, to minimize schedule impacts and evaluate productivity.

The previous paragraphs describe how the measures for "Schedule and Progress" were selected. A similar method was used to select the measures in each category.

A1.1.3 Integrate Into The Software Process

At the completion of the measurement selection activity, the IPT has defined a prospective list of measures, along with perspective data items, attributes, and aggregation structures. The next activity is for the developer to define a detailed measurement specification for each selected measure. The results are documented in the Project Management Plan. The detailed specification for the lines of code measure is provided in Table A.3.

Measure	Lines of Code (LOC)
Data Items	Number of LOC Number of LOC Added Number of LOC Modified Number of LOC Deleted
Attributes	Data Type (plan, actual) Data Collection Date Data Reporting Date Organization Source (new, reused) Language (Ada, C+, Assembly) Version Number
Structure	Component by CSCI
Definition	LOC will be counted as logical lines of code. No blank lines or comments will be included.
Collection Level	SU
Actual Count Is Based On	A CSCI is counted as complete when it passes CSCI qualification test. This means that code is complete and turned over to CM, the SU Test phase testing is complete, and code inspection and all outstanding action items from the inspection are complete.
Applied During	Estimates are calculated during software requirement analysis and design. Actual data is available during implementation. Actual updated data is re-measured during system integration and test, if a CSCI is modified to integrate a fix.
Data Reporting Process	SU-level data is available from the CM system used on the project. The government may access this system at any time to do detailed analysis. The government is provided a CSCI-level report of this data once a month via electronic format.
Frequency	Monthly

Table A.3 Specification For Lines of Code

All decisions made in this tailoring example were documented in the TMP including:

e. The list of project-specific issues, along with associated details describing each issue.

f. TPM common software issue and category to which each project-specific measure maps.

g. The measures selected to address the issues and the rationale for selecting each.

h. The measurement specification for each selected measure.

i. List of interfaces.

The measurement plan was implemented on this project. Data was collected and analyzed monthly. The PM used the analysis results.